|  |  |  |
| --- | --- | --- |
| **Validator Name** | **Validators** | **Sanitizers** |
| Joi | Binary, Boolean, date, number, string, email, array, credit\_card, ipAddress, ranges for numbers, etc. |  |
| First a Joi schema, schema is created using Joi.object. Then, some data object follows the schema  structure is validated using schema.validate(data). An error is returned if the validation fails.  ● A schema can also be a JavaScript object with each of the keys being Joi types. When such an object is passed to schema.validate, then it converts the object directly to a Joi type with keys.  ● Joi schema objects are immutable. This means that any new rule or key added to a Joi object will return a new schema.  ● It supports Strings, Numbers, Boolean, etc. and also validators for email, regex patterns, ranges for numbers.  ● It supports Validation against Data References. Eg- If we have 2 fields, wealth and savings in a schema, then savings can never be > wealth. This condition can be implemented using the ref() method.  ● It supports Conditional Validation. For example, fields related to credit cards will be shown only when the creditCardIsTrue field is true.  ● Joi also supports Conditional Relationships between 2 or more fields. Eg with(A, B) [Both A and B fields must be present] / xor(A, B) [Either A or B can be present].  ● One more thing that Joi can do is that it can be used to validate the request body of an HTTP POST request. [6] | | |